

**18-732**

**In-Class Assignment**

**Rohit Vaswani**

**Andrew ID: rrv**

format.c

1. A format string vulnerability is when a user controller input can be used as a format string parameter in C functions like printf and sprintf et al.

In the given code,

```
doit(char *str)
{
char text[0x100];
strncpy(text, str, 0xff);
text[0xff] = 0;
printf(text);
}
```

The first 0xff bytes of buffer str are used as a format string parameter to printf directly.

Thus, str, since is user controlled can be used to

- Crash the program
  - View large parts of the stack
  - Read from arbitrary memory location
  - Write to arbitrary memory location, among other things.
2. The important format string parameters are
    - %d – prints 4 byte aligned decimal number
    - %s – dereferences current memory content and prints it till the NULL character
    - %x – prints 4 byte aligned hexadecimal number
    - %u – Prints 4 byte aligned unsigned decimal number
    - %c – Prints 1 byte ascii value
    - %n – dereferences current memory content and writes the number of bytes so far

3. The stack trace is as follows

Grows downward towards lower memory addresses

---

Text – Buffer

---

Padding of 12 bytes

---

0xFF – arg of strncpy on stack

---

Pointer to str – arg of strncpy on stack

---

Pointer to the text – the format string

---

R.A to function do-it

---

4. `./format AAAA_%08x_%08x_%08x_%08x_%08x_%08x`  
Each `%08x` pops out data on the stack above the pointer to the format string.  
We continue doing this till we reach the starting of text Buffer on the stack
5. `./format AAA.%s`
6. Fix it with the function `do-it` having  
`printf("%s",text);`  
instead of the  
`printf(text);`